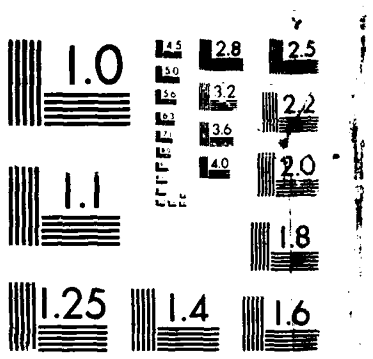
AD-A195 505 FEASIBILITY STUDY OF A KNOWLEDGE BASED FINITE ELEMENT MODELING ASSISTANT..(U) CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF CIVIL ENGINEERING.. G TURKIYYAH ET AL. 1/1

UNCLASSIFIED FEB 88 R-88-168 AFOSR-TR-88-0659 F/G 12/9 NL


1.0
4.5
5.0
5.6
6.3
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) R-88-168	5. MONITORING ORGANIZATION REPORT NUMBER(S) AFOSR-TR- 88-0659
6a. NAME OF PERFORMING ORGANIZATION Steven J. Fenves (OVER) Carnegie Mellon University 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION AFOSR
6c. ADDRESS (City, State, and ZIP Code) 5000 Forbes Avenue Pittsburgh, Pennsylvania 15213-3890	7b. ADDRESS (City, State, and ZIP Code) Same as 8c.
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Air Force Office of Scientific Research 8b. OFFICE SYMBOL (If applicable) NA	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER AFOSR-87-0092
8c. ADDRESS (City, State, and ZIP Code) Bolling Air Force Base Bldg 410 Washington, DC 20332-6448	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. 61102F · PROJECT NO. 2302 · TASK NO. B1 · WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Feasibility Study of a Knowledge Based Finite Element Modeling Assistant (Unclassified)

12. PERSONAL AUTHOR(S)
Turkiyyah George; Fenves Steven J.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM 86,Dec 1 TO 87,Nov 30	1988 February	45

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Knowledge Based Systems Finite Element Modeling Analysis Assistant

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The objective of this study is to explore the technical feasibility and issues involved in a broad cooperative effort to develop a knowledge-based expert system (KBES) environment for finite element modeling and analysis assistance. The environment is intended to serve the dual requirements of providing specialization capabilities to individual organizations so that they can "customize" the KBES to reflect their own expertise and generalization capabilities to serve as a "community memory" of the discipline. This report presents an architecture for such an environment. The key elements of the architecture are: a hierarchical, multiple-views representation of the spatial and functional characteristics of structural systems; an explicit representation and manipulation of modeling assumptions which allows fine-grained control over model synthesis; and an evolutionary model development strategy whereby modeling decisions can be methodically modified or retracted after an analysis iteration to generate more appropriate models.

Keywords: Artificial Intelligence; mathematical [illegible]

(OVER)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Anthony K. Amos	22b. TELEPHONE (Include Area Code) (202) 767-4937 · 22c. OFFICE SYMBOL NA

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

6a. Civil Engineering Department

19. The overall organization is based on the blackboard control model. A set of domain knowledge sources post, under a control strategy, information on a global data structure: the domain blackboard. The control strategy is dynamically created by control knowledge sources that post on a separate blackboard: the control blackboard. A simple scheduling mechanism monitors both blackboards and manages their knowledge sources. Domain knowledge sources use resources during their operations. Resources are the means to specify static declarative information to the system as well as to integrate finite element programs in the architecture.

FEASIBILITY STUDY OF A KNOWLEDGE BASED FINITE ELEMENT MODELING ASSISTANT

George Turkiyyah

Steven J. Fenves

Final Report
to
Air Force Office of Scientific Research
on
Grant No. AFOSR-87-0092
(CMU Account No. 1-53129)

February 1988

Department of Civil Engineering
Carnegie Mellon University
Pittsburgh, PA 15213

Table of Contents

List of Figures

Chapter 1
Introduction

The finite element method has developed rapidly in the last two decades and has essentially become the standard method of analyzing the functional performance of structural and mechanical systems across a variety of disciplines. The availability of finite element programs with very high levels of sophistication in analytical modeling capabilities, together with automatic mesh generators and graphical modes of interaction, have made the method particularly attractive for routine use in these disciplines.

However, these capabilities have not been matched by systems that can perform the physical modeling and interpretation tasks that are required to successfully apply finite element analyses to engineering problems. One reason for this deficiency stems, of course, from the very nature of these tasks. They tend to be heuristic, problem and situation dependent, with no consensus on specific aspects of the process even among experts. Existing computer tools have been unable to capture, in a practical way, the knowledge needed to perform these tasks.

The advent of knowledge-based systems and other AI techniques has given us some means to tackle the problem of partially automating the modeling and interpretation tasks. However, early efforts in using knowledge based systems to perform these tasks have been narrow in scope, hand-crafted for specific structures, and have not produced systems that can scale up to the level needed by engineering analysts. It is the object of this study to explore the issues involved in the development of a comprehensive knowledge based finite element modeling system and to develop some tools that can form the basis of a general purpose framework in which the modeling and interpretation tasks can be cast.

1.1. Problem Description

The function of a structural system is to resist applied loads without failure or excessive deformations. The objective of structural analysis is to determine the response of a structural system so as to evaluate the performance of a structure against the applicable performance and failure criteria.

Structural analysis, like other mathematical analyses in the physical sciences, can be globally decomposed into the following three generic steps depicted in Figure 1-1.

Modeling. This is the process of formulating a mathematical model that reflects the reality of the problem at hand. Modeling can be further subdivided into physical and numerical modeling. Physical modeling is the *idealization* of the problem by extracting from the physical structure the aspects of behavior that are deemed essential. The analytical model thus produced is a precise and unambiguous statement of the problem, including the variables of the problem and the equations that relate them. Although a symbolic solution of the analytic equations might be possible in simple problems, typically the problems formulated are more general than can be directly solved and numerical models have to be resorted to. Various numeric modeling techniques, differing in generality and ease of application for particular problems are available (finite strips, boundary elements, finite elements, etc.). In structural analysis, finite element models are by far the most widely used.

Solution. This is the process of solving the equations defined by the numeric model. It is a computationally expensive process but, in the current state of the art, it has been largely automated. Many comprehensive commercial programs exist for the solution of a wide range of numeric models. However, some special problems might require additional decisions such as determining precision, adjusting increments, controlling convergence rates and choosing efficient algorithms.

Interpretation. The solution of the numeric model being just a set of numbers, there is clearly the need to relate the numeric results back to the physical problem that was idealized by the model. This process is closely related to the modeling step; in fact, it is roughly its inverse, and is subdivided into two steps. The first step, model response abstraction, consists of the numerical interpretations which detect characteristics, trends and properties of the solution, match them with expectations about the model behavior and justify the assumptions built into the model. The second step, physical response abstraction, consists of the physical interpretations which relate the model quantities to the physical structure. A particular idealization implicitly defines a set of expectations that should be mapped back from the analytical solution to the original structure. In a broader context, interpretation gives directions for changing, improving or refining either the structural system being investigated or the model that was used for the analysis, or both.

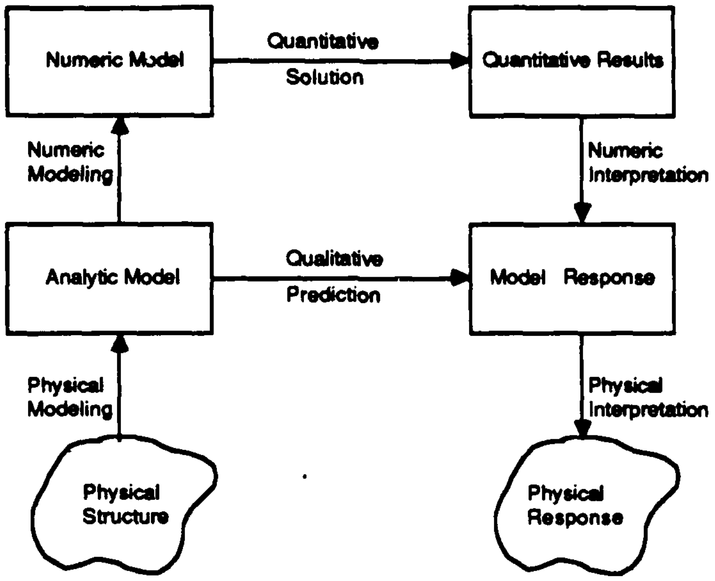


Figure 1-1: The Structural Analysis Process

It is the thesis of this work that:

- The user interaction with finite element analysis programs can occur at the *physical* level (i.e., in terms of problem features and performance criteria), without the need to deal with analysis details such as insuring the validity of particular idealizations, producing interpretations from analysis outputs, or learning the syntax of a particular finite element program.

- The framework that makes this interaction possible can be explicit enough so that it can be customized to particular organizations and uniformly expanded to model a wider range of physical phenomena.

- The physical modeling decisions can still be partly made by the user. The framework acts as an analyst's apprentice, allowing users to take control over the analysis to any degree they desire. Hence, the system represents an evolution in the way finite element analyses are carried rather than an addition of another black box in the toolkit.

1.2. Motivation

The motivation for developing a knowledge-based finite element modeling assistant comes from two sources: to extend analysis assistance into the areas of modeling and interpretation and to improve the linkage between CAD and analysis.

Assistance. Although finite element analysis programs have evolved since the early sixties [Fenves 63] to include more capabilities, more flexibility and better user interfaces, their basic structure and content remained essentially unchanged. They are still limited to the numerical level, they only accept very well defined and complete inputs and "for the most part, map a set of numbers given as input to another set of numbers given as output" [Agogino 87]. These characteristics make them not only difficult and time-consuming to use, but can also make them a potentially dangerous tool in the hand of inexperienced users because they hide the insights, ideas and assumptions that underlie those numbers. Smith shows many examples of mismodeling and systematically misleading outputs that can occur in using a finite element program [Smith 86]. Thus, a knowledge-based system could provide the following features to assist finite element programs users.

- Assistance in preparing input data. In complex structures, the amount of input required to use a finite element program is overwhelming[1]. A translator of higher level descriptions to finite element input can greatly reduce the turn-around time for an analysis and increase its reliability. Since finite element results are at most as reliable as the input given to the program, and mistakes are likely to occur due to the amount of input, enhancing the level of user interaction with such programs is highly desirable.

- Guidance to novice users in the choice of programs or options. The number and comprehensiveness of existing finite element programs can be bewildering to novice users, who, as a result tend to stick with sample procedures from users' manuals [Fouet 86]. A program that can explicitly deal with the objectives of a particular analysis task and the available program features can assist in the selection of an appropriate program or particular options of a given program.

[1]It has been estimated that 80% of the total time of an analysis is spent on preparing input.

- Identification of the limitations of a proposed model. The most common and dangerous errors in using a finite element program are not input errors, per se, but mismodeling errors. These errors are hard to track down and the inexperienced user might not be in a position to detect their presence. An analysis program that can represent the assumptions that led to a particular model from the physical structure can make it easier for a user to realize the shortcomings of a proposed model.

- Detection of errors in output. Finding errors in finite element results can be a very tedious activity due to the volume of the output. An ability to project expectations about the answers and match them with actual output numbers is another step towards facilitating the use of finite element programs and increasing their overall reliability. As a side effect, this feature provides the capability of stating the results in terms of more abstract response quantities.

- Overcome novice's tendency to complex models. The program can assist in generating the simplest model consistent with analysis objectives and available data.

Design. The second benefit of a knowledge-based analysis environment is its potential coupling with design programs. Analysis is rarely and end in itself: the overwhelming use of structural analysis is to guide and confirm design decisions. Unfortunately, current CAD environments suffer from two problems.

- They deal mostly with geometric design or spatial layout, with the analysis being relegated to an evaluation of design decisions previously made [Fenves 85]. The design is almost frozen regardless of the analysis results. A tighter integration is needed to allow for changes to be made, in a more interactive and routine manner, to early design configurations so as to steer the design towards optimality.

- The analysis facilities incorporated in CAD systems are usually used in a "blackbox" mode, whether they are part of larger analysis program or custom-written for the application. Such an approach not only lacks the potential of generalization to other application domains, but hides the idiosyncrasies of the analysis models and may thus obscure the performance limitations of the design. An analysis program that can reason about its models can provide a flexible interface to many applications and an explicit representation of the limitations of these models.

Moreover, generic knowledge-based structural analysis capabilities can be adapted to various paradigms of design illustrated below.

- Programs embodying a "hierarchical synthesis" philosophy need crude models at the early stages to eliminate competing alternatives as quickly as possible [Maher 84]. These models should evolve as the design progresses so as to take into account more features and project more realistic expectations.

- Programs embodying a "recall-tweak" philosophy are usually confronted with the problem that recalled designs have only the final specifications of the product, with no reference to the processes and objectives that led to it [Maher 87]. Changing the design to fit a new situation should start by an analysis that evaluates its important features, uncovers the relevant aspects of behavior and expresses them at a proper level of abstraction. Such knowledge can then be used to guide the refinement process that adapts the design to a new problem.

1.3. Knowledge Based FEM: Issues and Objectives

Explicit representation and separation of structure, function and behavior. "Structure" refers to the geometric composition of the structural system; "function" refers to an abstract organization and decomposition of the structural system that defines its "parts" (subsystems and components) and the roles that these parts are intended to play in resisting and transmitting applied loads; while "behavior" refers to the force-deflection response of the components and subsystems and the manner in which they interact to collectively achieve the global functionality. A representation that makes an explicit separation between these three aspects of a structure has many advantages. First, it is the basis for separating and classifying various kinds of knowledge about structural systems. Second, it allows a systematic investigation of the behavior because all behavior models could potentially be produced since there is no implicit encodings of behaviors in the structural or functional descriptions. Third, it allows to use the knowledge-based system in various modes (e.g., from knowledge of structure and function, check the behavior; from knowledge of function and intended behavior, design the structure; from knowledge of structure, derive the behavior and "discover" the function).

Explicit representation and manipulation of assumptions built into models. A system that does not treat assumptions as part of the modeling and analysis task is bound to do a selection out of a pre-enumerated set of models and will not have the capability to check the adequacy of the model after an analysis. This reduces the use of the program to specific classes or sub-domains of structures as well as the reliability in using the program, because the premises of the analysis are left implicit. Assumptions are a key part of any modeling task and their implications on models is a fundamental component of a modeler's expertise.

Explicit mapping of the behavior model to a finite element model. Such an approach reflects how human experts tackle the analysis task. They do no start an analysis without first visualizing the behavior of the structural system they are considering. In a finite element modeling assistant, this "visualization" should be represented by a "behavior model" and the generation of numerical models should be conceptually treated as a subordinate process driven by the need to quantify the characteristics of the behavior model, rather than being the ultimate goal of the analysis. Further, such an approach has an advantage in the interpretation of numeric results, because the mapping of the numeric results back to the behavior model is implicitly defined by the finite element model generation process.

Hierarchical model evolution. The availability of a series of models of increasing refinement is desirable for at least three purposes: to produce crude models that can be quantitatively analyzed at early design stages and refined as the design progresses; to produce abstract models that represent more compact descriptions of the behavior from refined models; and to allow incremental growth of confidence in the models generated. The last item is especially important in complex structures, where it is rather unusual to come up with a perfect model on the first shot, and the usual strategy is to hierarchically refine an initial simple model by checking and retracting assumptions.

Expressiveness. Expressiveness refers to the ability of the vocabulary used to define concisely the structural system and its behavior. The vocabulary should refer to geometric compositions, functional organizations and behavioral modes at various levels of abstraction in an easily understandable way. The availability of domain-specific taxonomies (organization of the vocabulary) to capture these concepts is desirable because it allows more efficient and meaningful user interaction without the need for the user to answer an overwhelming number of questions about the problem at hand. It also gives more transparency to the knowledge incorporated in the system: modeling heuristics can be predicated on features and actions specified in terms of abstract objects.

Customizability. Modeling has often been compared to an art. The basic principles can be formally learned but their application is often acquired from experience. Different people use different styles, assumptions and guidelines, which could potentially conflict. This has a pragmatic implication on a system aspiring to perform expert structural analysis: the system should have the capability to add modeling heuristics and/or tune the use of the heuristics it contains so as to reflect particular "styles" of modeling. Adherence to a "closed architecture" would severely limit the usefulness, while the implementation of a "core" that can be subsequently customized greatly improves flexibility. A corollary to incremental customizability is the gradual degradation or improvement in performance that the system can exhibit with the retraction or addition of "custom" heuristic knowledge. In the absence of simplifying heuristics, the system can fall back on basic principles at the expense of producing more expensive models that reflect the most general behavior of the structural system under consideration.

Expandability. While customizability refers to the ability to tune the process so as to reflect a particular style in a specific domain, expandability is a much stronger requirement. It refers to the ability to add an application domain by mapping the language of that particular domain to more general concepts defined within the architecture. This requires that some of the concepts be represented in a domain-independent manner. It is very unlikely that functional organization concepts can be domain independent, but behavioral and possibly geometric representations seem to have a generalization capability.

Flexibility of control. Flexibility refers to the ability of the system to decide on the generation of particular models by reconciling the simplicity of the model with the credibility of the assumptions built into it. Typically, modeling strives to produce the simplest models that are deemed adequate for reflecting the behavior of the structural system. The definition of adequacy is often implicitly predicated on both the simplicity of the model and on the estimated validity of the assumptions; there are always assumptions that are not formally checked yet accepted, as they fall below a threshold defined by the potential usages of the analysis results. A knowledge-based system should exhibit this form of flexible control so as to produce realistic models without the need for explicit user guidance to monitor model evolution and refinement.

Explicit control of reasoning. An acclaimed characteristic of expert systems is their ability to explain the reasoning process underlying their problem solving behavior. This characteristic is especially important in a program aspiring to be an *assistant*. Unfortunately, the currently available explanation models that trace back the problem solving cycles a rule at a time are too fine-grained for a useful explanation. Maintaining a separate representation of the control plan in terms of the strategies, goals, stages of the analysis, beliefs in assumptions, etc., can provide a more useful communication channel and, as an important side effect, allows a more dynamic choice of sequences of actions and better management of control heuristics in response to problem solving situations.

1.4. Organization of document

The remainder of this report is organized as follows:

- Chapter two provides the background;
- Chapter three presents the key ideas of this work;
- Chapter four proposes an overall architecture for the system;
- Chapter five summarizes the research and suggests recommendations for future work;

Chapter 2
Background

This chapter is divided into two parts. The first part reviews some previous attempts to build structural analysis assistants, either as stand-alone systems or as part of larger design systems. We present and discuss these systems and lay the groundwork on which we can expose our proposed system. The second part presents some work in the general area of reasoning about physical systems including the geometric and functional representation of systems, assumption based reasoning, and coupling numeric and symbolic techniques. Many of these ideas have influenced the proposed architecture of our system.

2.1. Structural Analysis Assistants

We examined five expert systems that deal with structural analysis: SACON, FEASA, PLASHTRAN, FACS and CARTER. The first three are stand-alone systems, while the latter two are connected to finite element programs in a design context. We separate the presentation into two sections: in the first section, we present the functionality of these systems and in the following section we discuss the perceived shortcomings in their approaches.

2.1.1. Overview of Some Precedents

SACON. One of the earliest systems that attempted to deal with the structural modeling problem was SACON [Bennett 78]. Following the heuristic classification problem solving paradigm [Clancey 85], SACON abstracts a simple real-valued-parameter model of the behavior of the structural system analyzed and applies rules to the model to infer important features such as controlling stress, deflection and non-linear behavior. Another set of rules uses these features to recommend appropriate analysis options for the MARC finite element program. SACON was implemented as a backward-chaining rule-based system with all the knowledge heuristic in nature. Although it did attempt to use substructuring and some geometric concepts (e.g., one-dimensional vs two-dimensional problem), it relied on the user to provide all such information by means of appropriate key-words. SACON was meant to be a demonstration of the EMYCIN environment and was never intended to be put in production. It succeeded superbly in showing that production systems are indeed a general formalism to represent the heuristic knowledge of experts.

FEASA. A second effort to build a modeling assistant is FEASA [Taig 86]. Unlike SACON, which assumes that the user has already modeled the physical problem in finite element terms, FEASA's objective is to assist in the identification of the features of the physical problem and to guide the user in the physical modeling decisions. FEASA gives advisory recommendations about problem specification and high-level modeling (needed before data preparation begins) through an interactive question/answer session. The system negotiates the problem size, mesh size, degree of refinement, etc. with the user. FEASA is not connected to an actual finite element program. Even with a very large knowledge base (2000 rules) FEASA still shows shallowness of advice. As the author points out, the basic reason for its deficiency is the limited knowledge representation and structuring permitted by the SAVOIR shell in which the system is implemented. The system cannot, as a result, handle substructures or develop hierarchical models. The knowledge base of what essentially are empirical associations grows quickly to an unmanageable size. This leads to an uncustomizable system that lacks the depth to attack more interesting analysis problems.

PLASHTRAN. PLASHTRAN [Cagan 87] is an expert consultant and teaching aid for modeling plates and shells using the MSC/NASTRAN finite element code. PLASHTRAN is written in LOOPS and uses a data-driven, forward chaining control. It decides on the appropriate analytic model of plates and shells (e.g., thin, thick or solid) by considering the thickness to length ratio. It uses some information on structural irregularities (e.g., stiffeners, material non-linearities) to determine the appropriate analysis strategy (e.g., linear or non-linear). It operates in a question/answer mode and gives recommendations on possible numerical modeling schemes, but is not actually connected to the finite element code. PLASHTRAN has an update feature that allows the user to change any data item and have the recommendations and inferences automatically updated. It does this by searching through the rule-base every time a variable modifies its binding to determine new assertions. It does not keep explicit dependencies between data items and justifications of inferences.

FACS. FACS is a system under development to convert airplane designs to finite element models [Gregory 86]. The system is intended to produce finite element models and solve them, not just to give recommendations. The motivation for the project is that current finite element preprocessors are designed for the efficient conversion of geometric entities to finite element models when the dimensionality of the two are the same. They do not handle the case where dimensionality of the model is less than that of its geometry (e.g., a finite element preprocessor cannot produce plate elements from a 3-dimensional domain). Such a capability needs to use not only the geometric definition of the components, but expertise on the function and expected behavior of the components. Since the available guidelines and procedures that define this knowledge are expressed at a higher level of abstraction than the geometrical descriptions of points, lines and surfaces, the first need is to construct a vocabulary capable of expressing component types at a suitable level of abstraction.

FACS is composed of five modules that act sequentially:

- retrieve geometric data and break this data into structural components that form a higher level description;
- classify structural components into their types according to the functional roles they play in the system using a discrimination network;
- exercise a rule-based knowledge base to choose methods and parameters for the model;
- apply algorithmic routines to produce generic finite element meshes; and
- translate the generic mesh into the syntax of the target program.

CARTER. CARTER is an expert system for the analysis and dimensioning of casings for mechanical components such as engine transmission units. CARTER uses two levels of structural analysis. First, in a predimensioning phase, behavior models of standard geometrical shapes that best approximate the given structure are invoked from a library. These models are simple enough so that strength of material solutions can be used to estimate deformations and vibration frenquencies. Using these simple models, a casing structure is synthesized. Second, a finite element analysis of the structure generated is used to check its dynamic behavior, in particular to check whether the natural frequency is sufficiently separated from the working frequencies. The finite element program used was specifically implemented for the class of problems that CARTER solves, and CARTER makes implicit assumptions when invoking the program (e.g., uncoupled bending and membrane effects). CARTER does not have substantial capabilities to refine the analysis model when it is unsatisfactory, but it does have capabilities to use the finite element analysis results to guide the design process. This is done in two steps. A local step uses the first mode shape (normal to the surface) of a substructure to modify thickness and/or boundary conditions of that substructure. A global step uses the overall behavior characteristics (one-way or two-way bending) to give advice for overall rib partitioning. This latter step is not implemented to any substantial level.

Conceptually, CARTER has many interesting features. However, because the knowledge incorporated is very particular to crankcases of a particular class of mechanical components, it makes it very difficult to generalize due to the implicit assumptions that the system makes about shapes, behaviors, results, etc. What is needed is a way to express the knowledge embedded in CARTER at a higher level of abstraction, and then instantiate it for particular classes of structures, whether crankcases or other structural systems.

2.1.2. Discussion

Although the systems reviewed above have achieved some success in their various application domains, they leave out many desirable features of a structural analysis assistant as summarized below.

First, all the systems presented are patterned after the classification problem solving paradigm. Models are not constructed but selected. The systems effectively act as big switches for models and algorithms and are limited to narrow application domains and even particular problem instances. The systems use productions to fill the parameters of complete model templates based on a set of a-priori chosen features. They do not exhibit any explicit reasoning about the functionality of the structural systems they analyze and how the functionality is achieved by the interaction of components that individually behave in some fashion.

Second, the knowledge incorporated in these systems is mostly empirical in nature, representing the experiential knowledge of their developers. This makes it difficult to generalize the knowledge and would lead to an explosion of the knowledge base size as more cases are covered. Although, admittedly, there is an empirical component in modeling, there are other kinds of reasoning on top of which heuristics can be built and that can make the expression of those heuristics more compact. In particular, the exclusive use of heuristic knowldge leads to a system that can presumably solve difficult problems but stops short of reasoning about much simpler problems (on a human scale). Furthermore, the nature of the empirical knowledge is typically specific and inflexible, and does not reflect the fact that many chunks of the analysis knowledge are generic and are used by humans for various purposes including checking, evaluation, design, assessment, instruction, etc.

Third, the systems presented do not treat assumptions explicitly, even though assumptions about the behavior of the structural system are the basic entities from which models are constructed, evaluated and judged adequate. The lack of explicit treatment of assumptions is mainly due to the absence of a formalism to express those assumptions. Such an omission obscures the knowledge incorporated in these systems and raises questions about their reliability. Further, this omission leads to an inability to hierarchically refine the analysis models, because a single and complete model that implicitly makes many assumptions about the structure is instantiated. Thus, no reasoning on how individual assumptions affect the form of the numeric model and the eventual analysis results can be performed.

Fourth, interpretation capabilities are limited. In particular, the interpretation is either non-existent, treated as a unilateral post-process, or relegated to the design program that uses the results in a particular process. The capability to evaluate the chosen analysis strategy and integrate the interpretation of the model solution into another pass that refines the model is not addressed. In fact, the problems mentioned in the previous paragraphs compound, and make it very difficult to incorporate such feedback. We think that modeling and interpretation should not be treated independently: they should access the same information and the content of an interpretation should be defined during the generation of a model.

2.2. Reasoning about Physical Systems

In this section, we bring together some ideas and issues that influenced the conceptualization of the proposed system and are relevant to the presentation of the next chapter.

2.2.1. Representation of Physical Systems

We consider the representation of physical systems at two levels: geometric and functional.

Geometric Representation. The need to process geometric information is common to many applications including computer graphics, computer-aided design, computer vision, etc. and interest in representing and reasoning about shapes and spatial relations is increasing in the field of expert systems [Woodbury 87].

Currently, the representation of geometric models in expert systems lacks some important characteristics. First, the models are specific; this specificity is the origin of their success. They are incarnations of a single view of their information content and generally tuned to a single application. They are fixed schemas where parameter filling is the basic operation leading to a very rigid descriptive capability. Second, these models do not reflect the perceptual organizations that we impose on objects. The notion of what represents a "part" is an intuitive capability for humans that is not, in general, capturable by these models. Third, the complexity of these models quickly grows when describing composite objects, because the models cannot reflect the simplicity of an underlying structure that often exists. These last two items are important for a friendly and meaningful man-machine interaction.

Woodbury [Woodbury 87] proposes an architecture for geometric reasoning. His architecture includes four elements: classes of spatial sets, features, abstractions, and constraints. Classes are objects, in the object oriented sense, and together with their associated methods are used to represent the characteristics of and the allowable operations on solid objects. Features provide a labeling mechanism: they allow naming and accessing particular parts of a geometric object. Abstractions are partial representations of features. Algebraic constraints on features and abstractions provide the mechanism to express and enforce the relationships between the objects of a given domain.

Functional Representation. The spatial description of structural systems provides only a weak representation of these systems because the spatial representation does not provide insights into the underlying organization of the structural system. Engineered systems are designed to fullfill specific functions; substantial knowldedge about the roles of the various parts and their intended purpose is implicit in this organization. The representaion that encodes this organization is termed the functional representation.

In structural analysis, the functional representation reflects how the structure is conceptualized to resist the applied loads. The functional description of a structural system is represented along a horizontal dimension and a vertical dimension. The horizontal dimension corresponds to different views of the same structure (or substructure) as derived from the various functional roles it plays, i.e., how it resists various loads. We use the term functional models to refer to these different views. The vertical dimension correspond to hierarchical abstractions in the functional models of the structural system.

An example of the use of functional hierarchies in structural analysis is provided in HIRISE, an expert system for the preliminary design of high-rise buildings [Maher 84]. HIRISE uses two functional hierarchies: gravity and lateral. The top level lateral description represents alternate overall lateral functional models (e.g., tube, core). These are hierarchically decomposed into descriptions in two orthogonal directions (e.g, frames, walls), and then into components (e.g, beams, columns, diagonals). The top level gravity description representing the alternate overall gravity functional models (e.g., one-way reinforced concrete slab, two way concrete-topped steel deck) is again decomposed into components (e.g., slabs, beams, columns). Interactions between the lateral and gravity functional models is implicitly handled by the order in which these models are considered. An extension to HIRISE, ALLRISE [Sriram 86], included explicit interaction constraints to represent the fact that a component can play multiple roles in various functional models.

2.2.2. Reasoning about Assumptions

All structural analysis schemes make assumptions. In fact, it could be argued that structural mechanics knowledge is either knowledge of assumptions or expertise in mathematics (symbolic algebra and theorem proving). However, in most analysis systems these assumptions are deeply embodied in the program code and are documented only in user's manuals. Assumptions are generic to all disciplines that use mathematical models. It has been recently realized that explicitness in expressing, reasoning and evaluating assumptions underlying mathematical models is desirable and that it is feasible to start investigating the possibility of incorporating them explicitly in expert systems.

Wellman [Wellman 86] studied the impact of assumptions on the structure of models produced[2]. His illustrations were taken from the statistics domain, namely, multi-attribute decomposition. In particular his interest was in producing utility preference models of a system described by a number of real-valued parameters. The utility preference models are algebraic relations between the parameters. Starting from a set of assumptions about the inter-independence of these parameters, theorems are applied to generate the functional form of the algebraic relations. The interesting observation is that with the addition of assumptions he could construct simpler forms that capture the behavior of the utility function. A set of assumptions can generate many models. In Wellman's work a simple metric was used to compare models: the number of independent scaling constants needed to define a particular instantiation of the functional form.

Using such a framework in structural analysis is not straightforward. The axiomatic nature of multiattribute theory does not translate into structural mechanics. First, mechanical behavior assumptions cannot be simple predicates used as axioms. Second, the use the assumptions to generate structural models cannot be formalized in a few theorems. Third, the evaluation of a model (in terms of its simplicity and validity of the assumptions) is often difficult and is particular to every problem instance. However the conceptual organization of the knowledge is useful.

An explicit representation of assumptions in a structural analysis domain is given in PROMPT [Murthy 87]. In PROMPT, the addition and retraction of certain sets of assumptions[3] allows for

[2]Note that this is different than that of generating assumptions, which is a highly domain specific and depends on the particular problem solved and the goal of the analysis.

[3]In PROMPT, the use of the term assumption refer to design features (e.g., rectangular section) that can be retracted only by modifying the design, as well as behavioral assumptions (e.g., elastic material) that can be retracted to obtain a more refined analysis.

the navigation through a graph of models, which change the behavioral description of the component being modeled. However, in PROMPT all the models that can be potentially produced by the assumptions are static entities and there does not appear to be mechanisms to evaluate the goodness of assumptions after the analysis is performed.

Reasoning about assumptions not only involves the use of assumptions to generate models and the verification and interpretation associated with the assumptions, but also the ability to devise mechanisms for maintaining those assumptions. This latter task can be done by an assumption maintenance system (also termed truth maintenance, reason maintenance or belief revision), separate from the problem solver that uses those assumptions, which keeps track of the assumptions and their implications. The goals of an assumption maintenance system are threefold [deKleer 86]. First, it is a cache mechanism for the inferences made during problem solving. Second, it is a way of dealing with the non-monotonicity that often exists in a reasoning system by detecting inconsistencies and contradictions that might appear during problem solving. Third, it is used to support backtracking when an assumption is incorrect by undoing all the inferences dependent on such an assumption. Efficient systems are currently available to support the modular assertion and retraction of assumptions with all the book-keeping being transparent to the user of the assumption maintenance system [McAllester 82, deKleer 86].

2.2.3. Coupling Symbolic and Numeric Computing

As expert systems began to be applied to engineering problems, it was recognized that the coupling of the symbolic processes with existing numerical algorithms is inevitable. Standalone symbolic processes are insufficient, while numeric programs are difficult to use because they require judgement and care often not easily available. In order to solve complex problems in engineering both insight (as typically available in symbolic structures) and precision (as available from numerical algorithms) are needed. It is often the case that insight into the problem must be gained in order to obtain a numeric solution and interpret the numerical results.

The definition of what constitutes a coupled system is tricky since many expert systems employ numeric techniques to some extent. A system can only be termed coupled if it has knowledge of the numerical processes embedded within it, and can reason about the application or results of those numeric processes [Kowalik 86]. This definition eliminates MYCIN, for example, as a coupled system just because it computes and propagates certainty values. The degree to which reasoning about the application or results of numeric processes is performed, places a system on a coupling scale ranging from shallow to deep.

Shallow coupling refers to systems that treat numerical processes as black boxes such as the application to manage numeric routines for the solution of non-linear algebraic equations [Talukdar 83]. The system manages the application of three different numerical programs. However, because little is known about the algorithms, the sequence of application of the routines is determined by watching the effect of the routines on the convergence of the state variables of the problem. The term "shallow" should not be taken in a pejorative sense, since the most successful applications have employed shallow coupling so far and in some cases this is the only possible alternative.

Deep coupling, on the other hand, refers to systems that utilize knowledge about the numerical processes employed and integrate this knowledge with other information during problem solving. Numeric routines are not used by watching for their effect after they are applied but are purposefully applied because of their known or expected utility and the need for them in the solution process. Deep coupling, because it requires knowledge about the characteristics of the numeric processes, enables a more intelligent interface to numeric routines. Deeply coupled systems are better performing, robust, and easier to expand but suffer from disadvantages related to the time required for their development and the amount of knowledge that need to be explicitly represented [Kowalik 86].

The coupling of symbolic and numeric computing raises new issues related to the architecture that can combine both kinds of processes and to the kind of knowledge needed to support the application of law-like knowledge that the numeric processes presumably embody. Two programming concepts have been proposed for coupling [Kowalik 86]: a blackboard architecture where a global memory (the blackboard) serves as the communication medium between the processes, and an object oriented programming environment where record structures (objects) encapsulate individual processes and buffer the expert system from the details of those processes. Symbolic and numeric processes communicate with each other via messages.

The question of what kind of knowledge is needed in using law-like knowledge is much bigger in scope. Zytkow in [Kowalik 86] points out that although scientific knowledge is comprehended into relatively condensed chunks and is relatively well structured, little progress has been made in the use of such knowledge in expert systems. The fundamental difficulty comes from the fact that the mathematical formule used to denote scientific laws are only the tip of the knowledge-iceberg that constitute the environment in which laws can be interpreted, applied, given meaning, and associated with their range of application. Only within such an environment can numerical procedures be reliably coupled with symbolic procedures, since the most important information about a numerical procedure are the equations that it solves while equations have to be associated with a lot of information that relate them (as idealized constructs) to the real-world objects and processes they represent.

The coupling of symbolic and numeric computing is important to us because the utility of our system comes first and foremost from the coupling of modeling and interpretation symbolic knowledge with numeric finite element programs. In the current state of the art, finite element programs are largely monolithic FORTRAN programs and it is a challenge to represent the equations that the various program configurations can solve and relate these equations to real-world structures. Furthermore, to achieve robust deep coupling we need an abstract representation of what the numeric procedures do, what their inputs and outputs are, and what are the things that may go wrong when they are invoked.

Chapter 3
The Building Blocks

This chapter develops three ideas central to this investigation. First, the representation of structural systems is presented. We discuss the properties of our representations and show how they can satisfy certain requirements of the framework, namely that the structures are *constructible* (i.e., there is no commitment to particular geometries), can be defined at multiple levels of abstraction, and are hierarchically organized into functional elements that play specific structural roles.

Second, behavioral assumptions are presented and shown to be the fundamental element of a knowledge-based modeling system. We develop the notion that any analysis model could be generated by the systematic application of a particular set of behavioral assumptions, and show how a generative problem solving method, based on deliberate commitment to assumptions, provides the generality needed to construct a variety of models that are useful during the analysis of a structural system.

Finally, we discuss how assumptions can be *reflective* knowledge structures, i.e., they can be designed to reason about themselves, by virtue of comprising post-analysis evaluation demons. Hence, modeling decisions can be methodically modified or retracted after an analysis iteration to produce more refined or elaborate models whose solution provide the quality of response appropriate to the analysis goals. This characteristic provides the system with the robustness and transparency necessary in the "core" of a finite element modeling assistant.

3.1. Representing Structural Systems

Finite Element Programs are "blind" to the physical phenomena they analyze. This fact results in serious problems ranging from the inability of the programs to successfully solve a problem to getting results that are plain wrong. Imparting the capability of reasoning about the object of the analysis to the programs requires first the ability to represent the system that is to be analyzed. This representation, which intent is to capture an "understanding" of the structural system, includes the spatial representation and the functional organization of the structural system as well as the representation of analysis goals.

This section discusses:

- The capabilities of the communication channel through which the user describes the structure. What should the capabilities of the system be at the descriptive level, so that it can be used effectively?
- The internal representation of the spatial, functional and other information. What about a structural system should be represented to be able to analyze it?
- The additional inferencing needed to complete the description of the structural system and start the modeling phase. This includes generating parts of various functional views and defining a "context" for the analysis, such as defining the applicable loading conditions.

3.1.1. The Spatial Model

There is a strong interaction between the spatial characteristics of a structural system and its behavioral or functional characteristics. Structural objects acquire their most characteristic behavior and function from their topological and geometric configurations. Spatial entities enter the design or analysis relations at all design stages. Hence the spatial representation of structural systems is of primary concern in this investigation.

One of the more important spatial problems that a FEM modeling assistant has to address is the granularity of the transaction that can happen between the user and the system. Choosing, a priori, to work at a particular level of geometric abstraction defeats the purpose of a knowledge based system. Such capabilities can be better implemented using traditional techniques, especially since an explicit a priori representation of the spatial characteristics and features of the problem meshes very well with the idea of parametric finite element models.

The interesting issues, from a research point of view, come when the user can communicate with the system at various levels of abstraction and the models that are generated can have a different level of spatial abstraction than that of the descriptive level (the level at which the user communicates the description of the structural system). The representations should be designed to: allow for the nonuniformity of spatial abstractions between different parts of the structural system (hence the need for a non-manifold representation); keep track of the more detailed geometric definition (or redefinition) of the structural system as it becomes available; and handle the shift in abstraction level that will occur during modeling.

Dealing with complex physical systems entails the need to view them as more than just the instantiation of templates. In particular, the description of a structural system should be in terms of a set of commands for building the structure, with no a-priori constraints on the possible ways the primitives can combine. This is crucial for generality. For example, the rectangularity of some unit is a particular feature that might turn out to allow for simpler models. But such a decision should be an explicit one and the system should not, in its core, rely on that feature, or any other, to do inferencing. Hence, a representation of spatial objects should have primitives, means of instantiating, transforming and combining these primitives, as well as means of defining new primitives that can be operated on by the above operations.

The spatial representation in this study defines three primitives: joints, members and panels. Intuitively, joints are objects whose dimensions are small with respect to the rest of the system; members are objects that are relatively long in one direction; and panels are objects whose length and breadth of are of the same order of magnitude but the thickness of which is considerably smaller. The locations of spatial objects are defined by a homogeneous transform matrix (T6) describing the position and orientation of the object relative to some other frame of reference. An object is instantiated by defining parameters and a location matrix for it. Objects can also be transformed (stretched, tapered, curved, rounded, etc.) by defining a transformation matrix that multiplies all its elements as defined in their natural coordinate system.

Two constructive operations are provided: a combination operation and a removal operation. One can define new parameterized templates from the primitives by instantiating and combining them. For example, to define a rigid frame of one can instantiate n beams, m columns and specify that they should be connected rigidly at their intersections (n and m can be parameters that will take values when a particular rigid frame is instantiated).

There are three "levels" of combining objects:

- The objects are essentially independent: they act separately, except that if they happen to touch they can transmit reactive forces.
- The objects are linked so that they can not separate: forces, but not moments, can be transmitted between them.
- The objects are integrated: the intent is to make them act as a single continuous body.

In the case where intersections of objects occur at physical joints, the semantics of the different levels of combination are clear. These semantics, however, are substantially more general and can handle many other patterns of intersections. For example, they can differentiate between the ways in which beams or girders interact with floor slabs, the ways in which an inside core interacts with outside frames, or the ways in which two beams on top of each other interact.

It is to be noted here that these spatial entities are *abstractions*, e.g., it is not necessary that a panel represent a wall or a slab: it could be the enclosing boundary of a frame. As the spatial model expands, the meaning of the spatial elements gets redefined. In fact, the meaning of the spatial abstractions is derived from another knowledge source that contain information about the application domain and the meaningful objects of that domain. The structural system gets represented as a set of functional hierarchies. This is discussed next.

3.1.2. Functional Models

A structural system has a unique spatial extent. It can however fulfill various roles. These roles (functional views or models) give semantics to the spatial objects. Their representation and use offer many advantages:

- First, functional models reduce the complexity in analyzing structural systems because those models impose a hierarchical conceptual organization on the geometric model.
- Second, functional models allow a more meaningful user interaction because the primitives of the functional description are the entities that engineers commonly use to describe structural systems. In fact, the library of spatial templates that are defined are likely to correspond to specific functional components (e.g., in buildings we might have rigid frames, K-braced frames or staggered wall-beam systems).

- Third, functional models are pointers to a large body of heuristic knowledge about the behavior of a structure and the assumptions that might be applicable. Knowledge of the purpose of a component has a direct impact on the way that component should behave to achieve its purpose.

- Fourth, functional models provide the link to synthesis and design programs because in the early stages of design it is often the case that information about functionality is available before the structure is specified and it is the analysis results that guide the design process. Functional models are the natural place to store the abstractions of analysis results to compare them to the requirements that the structure should satisfy.

Functional taxonomies are defined using a three-layered approach. The top level defines the functional elements of an application domain. For example, in the buildings domain the relevant elements may be the abstract objects lateral and gravity load-resisting systems, and physical objects such as frames, tubes, cores, floors. The next level defines the concepts that are more epistemological in nature and can underlie other applications. Functional objects can be abstract objects, physical prototypes, instances, sets or partitions[4]. Slots of functional objects can be attributes, roles (e.g., resist, distribute or transmit forces), or parts. The bottom level is the implementation level and is defined by the schemata language used. Schemas have slots and fillers. The schema language allows for inheritance, demons, and associating meta-information to any part of the schema.

Recognizing and instantiating the relevant functional objects involves two complementary processes:

- When defining spatial objects, the user can include a `type` attribute whose value is one of the possible functional objects known in the domain. It is natural to expect that some of the subobjects that the user describes are functional objects.

- Every functional object has associated with it a "grammar" (encoded as a set of productions) that, when run, can anchor the functional object to the corresponding pieces in the spatial model. Because functional views are hierarchical and are mutually constrained by the spatial model, additional functional objects can be triggered until the structural system is "recognized".

[4]Partitions are subsets that share same parameters. For example, they could be used to represent all the similar beams in a frame.

Loading conditions are associated with top level functional objects. For example, in the buildings domain wind and earthquake loads are part of the description of the lateral object while vertical loads are part of the gravity object.

3.2. Assumptions and Model Generation

The "core" portion of a finite element modeling assistant has to address the following concerns:

1. how to generate the right analysis model
2. making the generation process explicit
3. making the generation criteria explicit
4. how to recover from the wrong model
5. how to expand the current model

In this section we discuss how our framework provides mechanisms to handle the first three items. The next section will address the last two.

3.2.1. Generation of Assumptions

We think that it is neither practical nor desirable to list all the possible models of a domain in a finite element modeling assistant. Instead, we are suggesting that behavioral assumptions be used as the generators of the models and view modeling as a heuristic search for the "best" set of assumptions. We offer the following justifications for our point of view.

As mentioned in chapter two, previous approaches to finite element modeling have selected a model from a fixed set of models, each of which is applicable to some set of geometric and functional features and includes a "packaged" set of assumptions built into it, often not explicitly stated. Such a mode of operation usually result either in an oversimplification of the problem or in needlessly expensive models. The explicit manipulation of behavioral assumptions not only allows the identification of the premises on which the analysis is based, but also provides fine-grained control over the models produced, by tuning individual features and aspects of the model according to the specifics of the problem at hand.

In structural analysis, assumptions refer to behavioral constraints on the components and subsystems, and represent a variety of insights about the structural system's behavior (e.g., assumptions such as "core stiffness negligible", "uniform load distribution", "no out-of plane deformation"). Assumptions are the basic entities that determine the prediction capabilities and the limitations of a model. Furthermore, they can be expressed at a sufficiently abstract level to provide a compact and comprehendable set of criteria to evaluate a model, abstract away from a particular application domain, and interact with a user at a reasonably high level.

Modeling has often been compared to an art, yet acknowledged to rest on formal principles. An explicit treatment of assumptions can bridge the gap between heuristic knowledge and structural theory, because behavioral assumptions specify the essential properties of the structure and relate them to the models. Without this explicitness it is hard to define the capabilities of a modeling system because it is difficult to tell whether the models produced come about from individual heuristics directly built in the system or whether the models were derived by applying sound principles of more general applicability to the specific situation.

3.2.2. Representing Assumptions

An assumption has the following components:

- **Identifier and roles.** This is the "header" of an assumption. The identifier is the name used to refer to the assumption, while the roles are the parameters which vary from one use of the assumption to another but which have well defined purposes. The challenge in developing assumptions is, of course, to define roles in such a way that they can be applicable across a wide range of situations. For example, `ignore-torsional-deformations` could be applied to a single member, a subsystem, or a complete building if the role `what-resists-torsion` is used and is defined for each of the cases.

- **Applicability conditions and preconditions.** Applicability conditions are heuristic estimates of the validity of an assumption reflecting an a-priori credibility rating of it. Preconditions, on the other hand, are more rigid tests and have to hold true for the assumption to be considered. Preconditions are used to deal with interactions and possible orderings between assumptions.

- **Expansion.** This is the procedural interpretation of the assumption, i.e. how an assumption gets transformed into elementary operations on the behavior model.

- **Evaluation predicates.** These are expressions to be evaluated a-posteriori (after the numerical model is solved) to check the validity of an assumption.

Assumptions fall into one of three classes:

- **"Component" behavior.** These are the assumptions that can assign stress-strain relations to components, decide on the interactions between their various deformation modes (e.g., buckling), or assign force distributions or displacement patterns to them.
- **Lumping.** These are the assumptions that can join components together and decide on which degrees of freedom to retain or ignore.
- **Substitution.** These are the assumptions that can replace a substructure by another one that can exhibit similar behavior at the level appropriate to the analysis goal. These assumptions are similar to the familiar condensation techniques in their effect on the model.

3.2.3. Using Assumptions to Construct a Model

The application of assumptions results in a *behavior model*. The behavior model is the link between the structural system as a physical object and the finite element model as a numeric model. The behavior model represents the abstract mathematical problem that the finite element model approximates. The primitives of the behavior model are derived from basic concepts of structural theory and provide a program-independent formal description of the features of the model to be solved.

Our representation of the behavior model is a constraint network. The constraints are the relations between the generalized degrees of freedom of the behavior model representing the load-deflection relations of components and substructures and the equilibrium and compatibility relations that should hold between these components and substructures. Operations on the constraint network (e.g., adding or suspending constraints, establishing or breaking links, assigning values to parameters) reflect the physical modeling decisions during the solution of a problem.

Constructing a behavior model contains two subprocesses: choosing and instantiating the relevant assumptions. Choosing the assumptions to apply depend on the factors represented by their applicability conditions, e.g., whether they are compatible with the quality of results appropriate to the analysis goal, whether the spatial model is refined enough to allow their application, and whether the cost of the resulting analysis is acceptable.

Instantiation of an assumption can occur when its preconditions are satisfied. An assumption is instantiated by filling in the roles with the corresponding elements of the structure being analyzed. This creates an instance of the assumption specialized to the problem. The resulting instance is then executed and makes the necessary changes to the behavior model.

When enough assumptions have been instantiated to define completely and unambiguously the behavior model (i.e., the geometry, material properties, loads and boundary conditions of the model), an operationalizing procedure (that has access to an automatic mesh generator and to the syntax of a particular program) is ready to generate the input to a particular finite element program. The operationalizing procedure is currently under development.

3.3. Model Validation, Evaluation and Refinement

This section sketches how the results of an analysis could be interpreted to check the validity of the model and refine it if necessary. In particular, assumptions that are "component" behavior assumptions or lumping assumptions can be evaluated after the analysis to check whether they are satisfactory, while the evaluation of substitution assumptions require a more detailed analysis to be performed (i.e., without instantiating the assumption in question).

The assumptions that assign stress-strain laws are readily evaluated after the analysis results are known. It is possible to construct a linear sequence of material models (or at worst a tree with a small branching factor) that correspond to more refined behavior and check whether a component is acting in a particular regime.

The assumptions that make decisions about second order effects and the interaction of deformation modes can be evaluated by estimating the additional forces that result from applying the forces in one deformation mode to the distortions of another deformation mode. For example, an estimate of the P-delta effects in a column can be obtained by computing the additional moment that results from applying the axial force to the displaced shape of the column. If the interaction force is beyond some threshold then the next assumption to consider will be to include the geometric stiffness matrix.

The assumptions that assign some action (stress, force or moment) state or distribution in objects can be evaluated by applying the resulting displacements from the analysis to the locations where the action assumptions were made and evaluate the values of the actions that result. This is similar to the idea of secondary moments in trusses, but the heuristic is more general in nature. For example, if it is assumed that the core resist no loads (that is an assumption that the forces and moments in core are zero), then the displacements that result from an analysis not including the core could be applied to that core, to estimate the resulting forces and moments in it.

The assumptions that assign some distortion or displacement state to objects can be evaluated by applying the forces generated by the analysis to those locations where the distortion or displacement assumptions were made. For example, if one assumes that axial deformations in the beams of a rigid frame are to be neglected, then one could apply the axial forces that resulted from the analysis to those beams to get an estimate of the accuracy of the assumptions. Similarly to the above class of assumptions, this heuristic is more general in nature. For example, in the case where a frame is not perfectly planar but is idealized as a planar one, the idealization could be viewed as an assumption on the displacements of the nodes of the skewed bay(s) that they are parallel to the idealized direction. Evaluating this assumption can be done by applying the forces computed at the member ends framing into the skewed bay(s) to it and getting an estimate of the out-of-plane displacements. Formally, the assumptions of this class can be regarded as Lagrange multipliers. However, most finite elements packages do not have facilities that compute those multipliers, hence these computations will have to be done separately.

Lumping assumptions can be viewed as special cases of one of the above two cases and evaluated accordingly.

Some comments are in order:

- If an assumption was used because not enough parameters of the structure were known, it might happen that the evaluation can not be made until those parameters are chosen by the design process, i.e., the model can not be validated. The evaluation predicates are held until those parameters become available, at which time the analysis might not prove acceptable.
- Evaluations are a first order approximation of the errors that the assumptions introduce and in some cases can be just added to the original analysis results to produce an estimate of the more refined analysis.
- Many evaluations do require a separate (smaller) analysis to be performed. However, the additional confidence that they give in the analysis outweighs, in our opinion, the additional computational effort.

If one or more of the assumptions need retraction, the effects they had on the behavior model is undone. This can be achieved by a truth maintenance system that keeps track of the dependencies between the behavior model parameters and the generating assumptions. New assumptions can now be triggered and a more refined model can be generated.

Chapter 4
The STREX System

The previous chapter described the representations and the modeling mechanisms of our approach. Although these are the major components of a finite element modeling assistant, they must be embedded in a system that manages the overall system operations, controls the order in which tasks are performed, and provides other utility functions such as communicating with finite element programs. This chapter describes the architecture that we are developing.

4.1. Overview of the Problem Solving Architecture

Figure 4-1 shows the overall system architecture. It is partly based on the blackboard control model [Hayes-Roth 85] where a set of domain knowledge sources post, under a control strategy, information on a global data structure: the domain blackboard. The control strategy is dynamically created by control knowledge sources that post on a separate blackboard: the control blackboard. A simple scheduling mechanism monitors both blackboards and manages their knowledge sources. Domain knowledge sources use resources during their operations. Resources are the means to specify static declarative information to the system as well as to integrate finite element programs in the architecture. Aspects of the architecture are adopted from Fenves [Fenves 86].

Syntactically, control knowledge sources are the same as domain knowledge sources; the only difference is that they operate on objects stored on different blackboards. While the decisions on the domain blackboard are operations on the structural system and its models, the decisions on the control blackboard refer to problem solving operations that make the application of particular domain knowledge sources desirable. Domain knowledge sources are controlled indirectly because they respond to decisions made on the control blackboard. The architecture integrates domain and control problem-solving in a single basic control loop: at every cycle one knowledge source is chosen and applied, modifying a blackboard entry and triggering new knowledge sources to become candidates for execution at the next cycle. The knowledge source chosen at every cycle can either be a control or a domain source, i.e., control decisions compete with domain decisions as part of the solution process. Hence, the system can adapt its control plan to

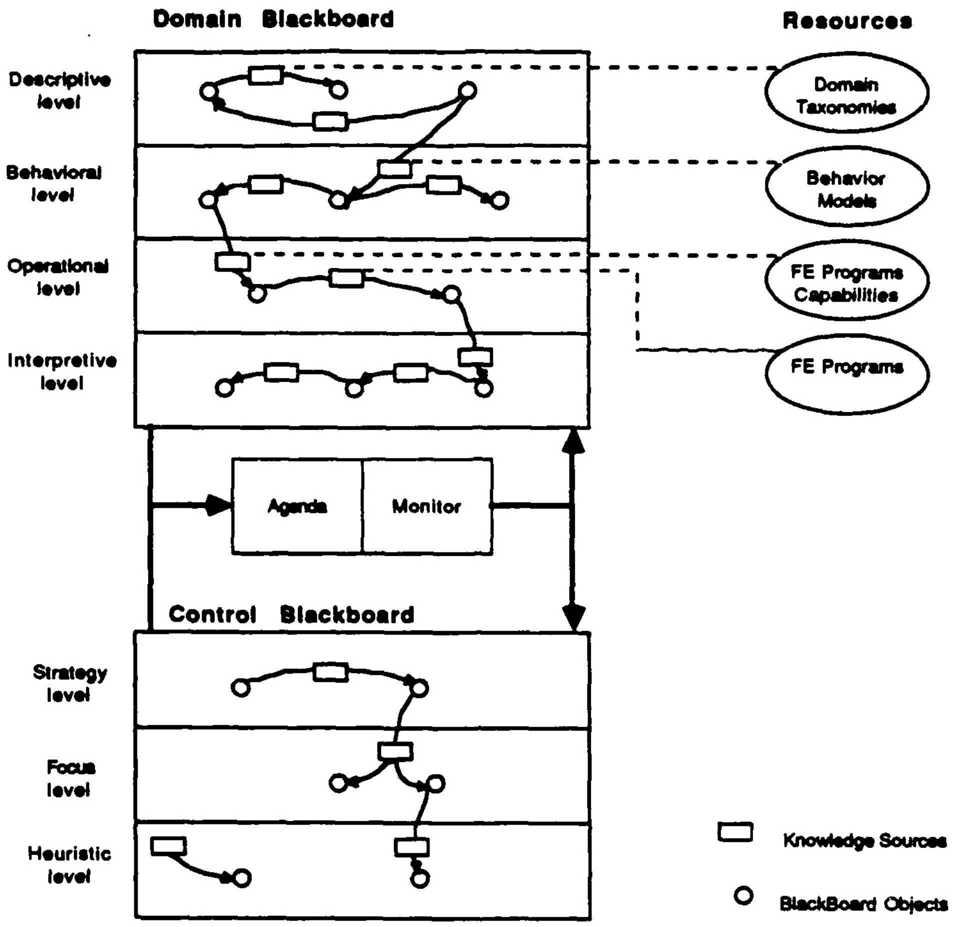


Figure 4-1: System Architecture

dynamic situations that occur during problem solving. Three domain independent control knowledge sources are responsible for scheduling: one updates the agenda at every cycle, another chooses a knowledge source to execute, and a third completes the cycle by transferring control to the chosen knowledge source for execution.

4.2. The Domain Blackboard

One way of visualizing the blackboard organization is along three dimensions. The "vertical" dimension along four levels: descriptive, behavioral, operational, and interpretive, represents the object description, the behavior projection, the analysis model specification and the interpretation of the analysis, respectively. The "horizontal" dimension reflects the multiple views imposed on the structural system by the fact that it is designed to resist multiple loading conditions. This dimension represents the various functional decompositions, at the descriptive level; the different behavior modes that the structure exhibits in response to the various loads, at the behavioral level; the various numerical models in the analyses, at the operational level; and the different resulting interpretations, at the interpretive level. The "out-of-plane" dimension represents the elaborations and refinements that occur during problem solving. This dimension represents the hierarchy in specifying objects, at the descriptive level; the evolutionary behavioral abstractions and idealizations generated, at the behavioral level; the numerical refinements that occur on the analysis model, at the operational level; and the evolving interpretations, at the interpretive level.

Using this three-dimensional metaphor, we can talk about blackboard levels (the horizontal planes representing different kinds of information about the structure), blackboard slices (the vertical planes representing the different views of the structure) and blackboard elaborations (the normal planes representing hierarchical refinements). While the blackboard levels are fixed for structural analysis problems and the blackboard slices fixed for a given structural analysis domain (buildings, bridges, automobiles, etc.), the blackboard elaborations are not bound a-priori. It is the interpretation at every step that guides the succeeding elaboration that dynamically occurs until the interpretation is satisfactory. A description of the domain blackboard levels, the knowledge sources that post on them and the resources that the knowledge sources use during their operation follows.

The descriptive level holds the geometric and functional hierarchies of the structural system. This is the object level at which the information about the structure being analyzed is represented. The interaction between functional subsystems is represented by the fact that some of their components are linked to the same geometric description. The descriptive level is built by two knowledge sources that fit two usage scenarios. First, in a top-down design context, a functional refinement KS develop the functional hierarchy "downwards" and link it to the geometric description. Second, in a design verification context, a geometry extractor KS derives suitable functional abstractions from a geometric description using a domain taxonomy resource that defines the various functional roles of structures and their subsystems.

The behavioral level holds models of the behavior of the structural system and its components, i.e., the generalized degrees of freedom and the load-deflection relations, as well as the relations between the hierarchy of models generated. Behavior models are built by knowledge sources that can generate, refine and abstract them using behavioral assumptions. The assumptions and the aspects of the behavior model that are entailed by the individual assumptions are maintained by an underlying assumption maintenance similar to ATMS [deKleer 86].

The operational level holds a numeric model for input to a finite element program for quantitative analysis. The finite element model is a prescriptive sequence of operations that specify (potentially after some translation from a neutral format) the procedures to be executed by the target finite element program. These operations specify the finite element mesh, the material properties, the loading history as well as the response type, the solution algorithms and output options. Two knowledge sources build the finite element model: the first KS chooses the relevant options of the finite element program that are needed to reflect the characteristics of the behavior model, and the second KS generates the commands to the program.

The interpretive level stores three kinds of evaluations that answer three types of questions: does the numeric model correctly solve what the behavior model represents? does the behavior model reflect the correct physical behavior? does the structural system satisfy its intended function? It is the information at this level that determines whether backtracking or refinement are needed and if so, whether it is the numeric model, behavior model or object model that should be changed. Three knowledge sources perform the evaluation of the numeric results, model response and physical response, respectively. The evaluation of the numeric results involves issues such as accuracy and convergence of the analysis results. The evaluation of the analytical model involves the validation of the behavioral assumptions and of the confidence in the appropriateness of the behavior model with respect to the goal of the analysis. The evaluation of the physical response is done by comparing key response parameters to design specifications and functional criteria such as stress levels, ductility requirements and deflection limitations.

4.3. The Control Blackboard

The control blackboard is organized along three levels: strategy, focus and heuristic, representing different levels of abstraction of control decisions. The decisions at these levels determine the control plan, i.e., the desirable actions to be taken. Control knowledge sources that control model generation, monitor model evolution and strategize design modifications post their decisions on the control blackboard. A description of the control blackboard levels and the knowledge sources that post on them follows.

The strategy level records the global system-level tasks that are to be performed. These global problem solving decisions depend on the input of the problem, the goal of the analysis and how far in the solution process the system is. We define four kinds of strategies that: abstract a functional model and then generate a behavior model; build a geometric model and then a corresponding behavior model; refine a behavior model; and refine an object model.

The focus level records local goals, objectives and interests that focus attention to domain knowledge sources or blackboard objects during the solution to implement the active strategies. We currently recognize four KSs that implement the four strategies presented above. Focus decisions determine, for example, the order in which to consider the various subsystems, the granularity of the models and the nature of the analysis.

The heuristic level holds control heuristics that, strictly speaking, fall outside the scope of structural theory but are useful to obtain a solution in reasonable time and/or using limited memory requirements. The control knowledge sources that post at this level include miscellaneous heuristics that assign weights to various control decisions.

Chapter 5
Summary and Status

5.1. Summary

In the introduction chapter we outlined nine objectives for a structural analysis environment. We believe that the proposed system takes a first cut at satisfying them. Some of the objectives are achieved by our concern for the principles, others follow directly from the organizational scheme proposed. We review these points briefly.

- Explicit representation and separation of structure, function and behavior is a direct consequence of the organization. At the descriptive level, a geometric composition is built and various functional models are linked to it. At the behavior level the various behavior models are separately represented.

- The explicit representation of assumptions built into models is achieved by requiring that the modeling operators communicate the assumptions they make to an underlying assumption maintenance system that records how the models built were entailed by these assumptions.

- Explicit mapping of the behavior model to a finite element model is done by blackboard actions that translate behavior models at the behavioral level to numeric models at the operational level.

- Hierarchical model evolution is achieved by the incremental activation and retraction of assumptions, since it is the application of these assumptions that has a simplifying or refining effect on the structure of the model.

- Expressiveness is achieved by the representation of functional models that are internally linked to their corresponding geometry. The elements of the functional models allow the description of a structural system to be conveyed by a few user-supplied abstract parameters.

- Customizability is achieved by constraining the heuristic portions of the system to well defined chunks (i.e., heuristics about behavioral assumptions, heuristics about the numerical solution and control heuristics), where they can be expressed at a relatively high level and modified, refined and augmented independently.
- Expandability is still a claim at this stage, but the representation of the behavior model in terms of common concepts of structural mechanics (e.g., bending, buckling, torsion, yielding) should make it possible to add new domains at this level. Clearly, more work will be needed especially for geometric primitives and functional organizations but we are hoping to generalize some guidelines for structuring functional models and linking them to their geometric descriptions.
- Flexibility of control is a consequence of using the control blackboard architecture and having separate knowledge sources that make independent inferences about the problem solving process.
- Explicit control of reasoning is another advantage of the control blackboard architecture. Process decisions are available on the control blackboard and a transcript of those decisions provides a user with the control flow.

One advantage of our approach, we feel, is that the modeling problem is conceptually treated with a formative problem solving method. Models are *constructed* under various behavioral assumptions. The architecture treats assumptions as heuristic constraints that can be activated and disregarded according to the problem solution. This approach gives more flexibility to the model generation capabilities than a classification problem solving method that essentially treats modeling as a black-box task.

It might be argued that the blackboard architecture is computationally expensive and imposes severe storage requirements on the system. This overhead, coupled with an already resource-intensive finite element program could tremendously decrease effective system performance. While we believe that this argument is true, we are using the blackboard control framework as the development environment and not as support for a high-performance, deliverable system. The architecture supports a very modular and flexible domain and control knowledge organization and will allow us to experiment with various ideas in an incremental fashion, prior to any finalization of a product. If the prototype turns out to be successful, a production version, guided by the features of the prototype, will reimplement the solution in a more efficient environment.

5.2. Status

The project was undertaken on the assumption that it would be completed in two years. This report summarizes the achievements of the first year, during which the major conceptual framework was formulated. Exploratory pre-prototype implementations of portions of the system were performed, including a small early pre-prototype which generated an operational model for execution by the SAP80 program.

It is intended that the second year be devoted to implementing the full STREX system described, with operational linkage to one or possibly two FEM programs. The specific "knowledge base" of the initial system (functional and behavioral models and assumptions) will be geared to the domain of building structures. However, consistent with the overall motivation of the project, care will be taken to build up a "core" modeling system expandable and customizable to other domains.

References

[Agogino 87]	Alice M. Agogino. AI in Design: Qualitative Reasonning and Symbolic Computation. 1987.
[Bennett 78]	James Bennett et. al. *SACON: A Knowledge-based Consultant For Structural Analysis.* Technical Report STAN-CS-78-699, Stanford Heuristic Programming Project, 1978.
[Bobrow 85]	Daniel G. Bobrow (editor). *Qualitative Reasoning about Physical Systems.* MIT press, 1985.
[Brachman 79]	Ronald J. Brachman. On the Epistemological Status of Semantic Networks. *Associative Networks: Representation and Use of Knowledge by Computers.* Academic Press, New York, 1979, pages 3-50.
[Cagan 87]	Jonathan Cagan and Victor Genberg. PLASHTRAN: An Expert Consultant on Two-dimensional Finite Element Modeling Techniques. *Engineering with Computers*, 1987.
[Clancey 85]	William J. Clancey. Heuristic Classification. *Artificial Intelligence* 27, 1985.
[deKleer 86]	Johan de Kleer. Problem Solving with the ATMS. *Artificial Intelligence* 28, 1986.
[Dyer 86]	Michael G. Dyer, Margot Flowers and Jack Hodges. EDISON: An Engineering Design Innovation System Operating Naively. In D. Sriram and R. Adey (editor), *Applications of Artificial Intelligence in Enginnering Problems.* Springer-Verlag, 1986.
[Ebner 75]	Alan M. Ebner (editor). *Task Committee on Automated Analysis and Design, Committee on Electronic Computation.* ASCE, 1975.
[Falconer 84]	W. D. Falconer and L. S. Beedle. *Classification of Tall Building Systems.* Technical Report 442.3, High-Rise Institute Report, Lehigh University, 1984.

[Fenves 63] Steven J. Fenves.
STRESS: A Computer Programming System for Structural Engineering Problems.
May, 1963.

[Fenves 85] Steven J. Fenves.
A Framework for a Knowledge-Based Finite Element Assistant.
In Clive L. Dym (editor), *Applications of Knowledge Based Systems to Engineering Analysis and Design*. ASME, 1985.

[Fenves 86] Steven J. Fenves.
A Framework for Cooperative Development of a Finite Element Modeling Assistant.
In K. J. Bathe and D. R. J. Owen (editor), *Reliability of Methods for Engineering Analysis*, pages 475-486. Pineridge Press, Swansea, U.K, 1986.

[Fintel 71] ACI Committee 442.
Response of Buildings to Lateral Forces.
Technical Report 442R-71, ACI, 1971.

[Fouet 86] Jean-Marc Fouet.
Artificial Intelligence in Structural Analysis.
1986.

[Garrett 87] James H. Garrett Jr. and Steven J. Fenves.
A knowledge Based Standards Processor for Structural Component Design.
Engineering with Computers , 1987.

[Garvey 86] Alan Garvey et. al.
BB1 User Manual - Common Lisp Version
Knowledge Systems Laboratory, Stanford University, 1986.

[Gregory 86] B. L. Gregory and M. S. Shephard.
Design of a Knowledge Based System to Convert Airframe Geometric Models to Structural Models.
In C. N. Kostem and M. L. Maher (editor), *Expert Systems in Civil Engineering*. ASCE, 1986.

[Hayes-Roth 85] Barbara Hayes-Roth.
A Blackboard Architecture for Control.
Artificial Intelligence 26, 1985.

[Kowalik 86] J. S. Kowalik (editor).
Coupling Symbolic and Numerical Computing in Expert Systems.
North-Holland, 1986.

[Maher 84] Maher, M. L.
HI-RISE: An Expert System for the Preliminary Structural Design of High Rise Buildings.
PhD thesis, Civil Engineering department, CMU, 1984.

[Maher 87] Mary Lou Maher and Fang Zhao.
Using Experience to Plan the Synthesis of New Designs.
In John Gero (editor), *Expert Systems in Computer-Aided Design*. North-Holland, February, 1987.

[McAllester 82] David A. McAllester.
Reasoning Utility Package user's manual.
AI Memo 667, MIT, 1982.

[Meyer 84] Christian Meyer and John McCormick.
Mathematical Modeling Of Complex Structures For Dynamic Analysis.
Computers and Structures 18(4):673-688, 1984.

[Murthy 87] Seshashayee S. Murthy and Sanjaya Addanki.
PROMPT: An Innovative Design Tool.
In *AAAI 87*. AAAI, 1987.

[Neuss 83] C. F. Neuss and B. F. Maison.
A study of Computer Modeling formulation and special analytical procedures for Earthquake Response of Multi-Story Buildings.
Technical Report Prepared for the National Science Foundation, CEE7926734, J. G. Bouwkamp, Inc., 1983.

[Reynier 86] Marie Reynier.
Interactions between Structural Analysis, Know-How and Chain of Reasoning used by the CARTER Expert System for Dimensioning.
In K. J. Bathe and D. R. J. Owen (editor), *Reliability of Methods for Engineering Analysis*. Pineridge Press, Swansea, U.K, 1986.

[Smith 86] G. Smith.
The dangers of CAD.
Mechanical Engineering 108(T63-2), February, 1986.

[Sriram 86] Duvvuru Sriram.
Knowledge-Based Approaches for Structural Design.
PhD thesis, Civil Engineering department, CMU, 1986.

[Struss 87] Peter Struss.
Multiple Representation of Structure and Function.
In John Gero (editor), *Expert Systems in Computer-Aided Design*. North-Holland, February, 1987.

[Sussman 80] Gerald J. Sussman and Guy L. Steele.
CONSTRAINTS-A language for expressing almost hierarchical descriptions.
Artificial Intelligence 14:1-39, 1980.

[Taig 86] Ian C. Taig.
Expert Aids to Finite Element System Applications.
In D. Sriram and R. Adey (editor), *Applications of Artificial Intelligence to Engineering Problems*. Springer-Verlag, 1986.

[Talukdar 83] S. N. Talukdar, S. S. Pyo and A. Elfes.
Distributed Processing for CAD - Some Algorithmic Issues.
Technical Report EDRC, CMU, 1983.

[Wellman 86] Michael P. Wellman.
Reasonning About Assumptions Underlying Mathematical models.
Coupling Symbolic and Numerical Computing in Expert systems.
North-Holland, 1986, pages 289-338.

[Woodbury 87] Robert F. Woodbury.
The Representation and Manipulation of Geometric Information in a Knowledge Based Expert System.
PhD thesis, Civil Engineering department, CMU, 1987.